

Maximal Intersection Queries in Randomized Input Models

Benjamin Hoffmann*

Universität Stuttgart, Germany

Mikhail Lifshits†

St. Petersburg State University, Russia

Yury Lifshits‡

California Institute of Technology, USA

Dirk Nowotka§

Universität Stuttgart, Germany

Abstract

Consider a family of sets and a single set, called the query set. How can one quickly find a member of the family which has a maximal intersection with the query set? Time constraints on the query and on a possible preprocessing of the set family make this problem challenging. Such maximal intersection queries arise in a wide range of applications, including web search, recommendation systems, and distributing on-line advertisements. In general, maximal intersection queries are computationally expensive. We investigate two well-motivated distributions over all families of sets and propose an algorithm for each of them. We show that with very high probability an almost optimal solution is found in time which is logarithmic in the size of the family. Moreover, we point out a threshold phenomenon on the probabilities of intersecting sets in each of our two input models which leads to the efficient algorithms mentioned above.

1 Introduction

The *nearest neighbor* problem is the task to determine in a general metric space a point that is closest to a given query point. This kind of queries appear in a huge number of applied problems: text classification, handwriting recognition, recommendation systems, distributing on-line advertisements, near-duplicate detection, and code plagiarism detection.

In this paper we consider the nearest neighbor problem in a “binary” form. Namely, every object is described as a set of its features and similarity is defined as the number of common features. In order to construct an efficient

*Email: hoffmann@fmi.uni-stuttgart.de

†Email: lifts@mail.rcom.ru

‡Email: yury@caltech.edu

§Email: nowotka@fmi.uni-stuttgart.de

solution some assumptions should be added to the problem. Here we assume that the input behaves according to some predefined distribution. Then we construct an algorithm and show that the time complexity and/or the accuracy are reasonably good *with high probability*. Here we use the probability over the input distribution, not over random choices of the algorithm. This probabilistic approach was inspired by the recent survey of Newman [18]. He gives a comprehensive survey about random models of graphs that agree well with many real life networks, including Web graphs, friendship graphs, co-authorship graphs, and many others. Hence, we can attack the nearest neighbor problem in already “verified” random models.

The Maximal Intersection Problem. Consider a family of sets and a single set. We ask for a member of the set family which has a maximal intersection with the query set.

The Maximal Intersection Problem (MaxInt)

Database: A family \mathcal{F} of n sets such that $|f| \leq k$ for all $f \in \mathcal{F}$.

Query: Given a set f_{new} with $|f_{new}| \leq k$, return $f_i \in \mathcal{F}$ with maximal $|f_{new} \cap f_i|$.

Constraints: Preprocessing time $n \cdot (\log n)^{\mathcal{O}(1)} \cdot k^{\mathcal{O}(1)}$.
Query time $(\log n)^{\mathcal{O}(1)} \cdot k^{\mathcal{O}(1)}$.

Let us restate the problem in a graph theoretical notation which will allow a more convenient description of some applications of MAXINT later. A database is a bipartite graph with vertex set partition (V, V') such that $|V| = n$ and the degree of every $v \in V$ is at most k . A query is a (new) vertex v (together with edges connecting v with V') of degree at most k . The query task is to return a vertex $u \in V$ with a maximal number of paths of length 2 from v to u .

Our main motivation for studying MAXINT was problems like text clustering, near-duplicate detection or distribution of on-line advertisements. In these problems, the database mainly consists of natural language text documents. Therefore, we will deal in the rest of the paper with documents and terms instead of sets and elements. On the other hand, we want to stress that our ideas and algorithms can be applied to every input following our models. Note that in this work documents are not considered to be multisets of terms. But, as we will see in Section 2, we use the fact that every term in a document occurs with a certain multiplicity.

Results. In Section 2 and Section 3 we propose two new randomized input models for MAXINT, called the *Zipf model* and the *hierarchical scheme*. Assume that the terms of a query document are ordered by their frequency in the document collection. Now consider the probability curves for the two following

events with parameter q (Figure 1). *Any q -match*: there is a document in the random (according to our models) collection that has at least q common terms with the query document (the solid curve). *Prefix q -match*: there is a document in the random collection that has at least the first q terms (according to the order given by the term frequencies) of the query document (the dashed curve). Both curves have the similar structure: the probability is close to 1 for small q , but suddenly, at some “matching level”, it falls to nearly zero. Our main ob-

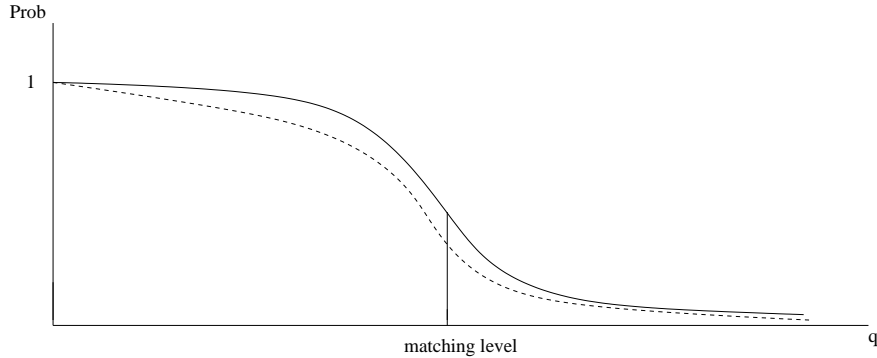


Figure 1: Exemplary probability curves for any q -match and prefix q -match

servation is that these matching levels for prefix q -match and any q -match are very close to each other. And this is extremely important for solving MAXINT. Indeed, finding the best prefix q -match is computationally feasible. We show that closeness of matching levels for any q -match and prefix q -match with high probability allows to find an approximate solution for MAXINT.

With respect to the conference version of this paper we generalize the matching level theorem from regular queries to any query taken from the Zipf model [10].

Applications. The MAXINT problem is a natural formalization for many practical problems:

Long search queries. Consider a bipartite graph representing websites and their words, that is, every website is represented as a set of words. Let a query be a moderately large set of words (say, 100 words). For example, one might get a large query by expanding a five word query by adding all synonyms. The search for a website containing ALL query terms might produce no result. Hence, a search for a website that has maximal intersection with the query set is a natural alternative. Therefore, efficient algorithms for MAXINT can help web search engines like google.com¹, yahoo.com, or msn.com to relax their restrictions on the query length.

¹The reference time for all links mentioned in this paper is April 2007.

Content-based similarity. Consider a bipartite graph representing documents and their terms. Finding a document in a database that has a maximal number of common terms with a newcomer document might be a basic routine for text clustering/classification and duplicates detection. Particular examples are news classification systems (reuters.com), news clustering (news.google.com), and spam detection.

New connection suggestions. Consider an undirected graph between people representing for example friendship or co-authorship. Here, every person is described by his name and a list of all his friends. Then, applying a MAXINT query to a (new) person we get a natural suggestion for establishing a new connection for her. Indeed, we get a person that has a maximal number of joint friends with the query person. Related systems can be found at linkedin.com (for acquaintances) and dblp.uni-trier.de (for co-authorship).

Co-occurrence similarity. Consider an audience graph between people and some items. Every item is represented by a set of people who are interested in it. Take a (new) item together with its audience. Then, the MAXINT query returns an item that has the maximal co-occurrence with the query item in people's preference lists. Particular examples are the music band similarity by their listeners (last.fm) and RSS-feeds similarity by their subscribers (bloglines.com and feedburner.com).

Advertisement Matching. Delivering advertisement relevant to users interests is one of the most important problems in web technologies [15]. MAXINT can reflect this challenge in a natural way: Consider a graph representing websites participating in some ad distribution system and their terms. A query is a set of terms that describe some advertisement and its target audience. Here, a solution of MAXINT suggests a website that is among the best candidates to display the given ad. See google.com/adsense as an example system for ad distribution.

Social recommendations. Consider a bipartite graph between people and their recommendations (e.g. for books, bars, cars). A query is a set of friends of some newcomer person. Finding an item that is already chosen by many of the newcomer's friends is a natural form of recommendation. The friendship graph together with recommended items is for example accumulated on facebook.com.

Note that for some of these applications the Jaccard similarity coefficient may also be an appropriate similarity measure.

Related Work. MAXINT is a special case of the nearest neighbor problem. Indeed, one just needs to define the similarity between two documents as the number of common words. There is also a way to define a *metric* (i.e. distance function satisfying the triangle inequality) providing the reverse similarity order.

To do this, we need to add some unique “imaginary” words to every document making their size equal and then use the Jaccard metric [2]. Denoting the maximal cardinality of a document in the collection by M the resulting formula is $d(A, B) = \frac{2M - 2|A \cap B|}{2M - |A \cap B|}$. Instead of the Jaccard metric one could use the size of the symmetric difference as well (again, one has to add unique words to every document making their size equal). This defines again a metric which provides the reverse similarity order.

Many efficient algorithms have been developed for nearest neighbor search in special cases or under various assumptions; see recent survey papers [1, 4, 5, 9, 11] and the book [19] for comprehensive reviews. Nearest neighbors are particularly well studied in vector models with the Euclidean distance function [13, 7]. Actually, we can interpret a document as a vector of 0s and 1s (1 means a term is contained in a document). Then, the scalar product is equal to the size of the intersection. Unfortunately, random projection methods studied are not directly applicable to MAXINT. Namely, (1) we do not allow that the complexity is linear in the vector length, and (2) a c -approximate solution for the Euclidean distance is not necessarily a c -approximate solution for the size-of-intersection similarity. Note that the length of the vectors (resulting from the overall number of different terms in the document collection) can be much larger than the size of the document collection.

Closely related to MAXINT is *text search*. Finding documents that fit best to some given search terms can also be considered as a problem on a bipartite graph. The documents and terms are the nodes and edges are drawn when a term occurs in a document. Basically the task is to find all documents containing *every* query term and rank these documents by relevance. The key technique in this area is inverted files (inverted indexing). A comprehensive survey of the topic can be found in [20].

2 MaxInt in the Zipf Model

Let $\mathcal{T} = \{t_1, \dots, t_m\}$ be a set of *terms* and $\mathcal{D} = \wp(\mathcal{T})$ be the power set of \mathcal{T} , called *documents*. A document collection \mathcal{D}_n is a subset $\{d_1, \dots, d_n\} \subseteq \mathcal{D}$. We demand $m \in n^{\mathcal{O}(1)}$. In the following we will use the terms *prefix match* and *any match* instead of prefix q -match and any q -match since the size of a matching is always stated explicitly. By \log we always mean \log_2 , while \ln denotes \log_e .

We now describe a probabilistic mechanism for generating a document collection called the *Zipf model*. Every document is generated independently. Term occurrences are also independent. A document contains term t_i with probability $1/i$. Hence, the expected number of terms in a document is approximately equal to $\ln m$ in our model. This model is similar to the *configuration model* ([18]) with Zipf’s law for distribution of term degrees and constant document degrees. Zipf’s law states that in natural language texts the frequency f of a word is approximately inversely proportional to its rank r in the frequency table, i.e. there exists a constant c such that $f \cdot r \approx c$ (Table 1). For more details about Zipf’s law see [17].

Word	Frequency	Rank	$f \cdot r$	Word	Frequency	Rank	$f \cdot r$
the	3332	1	3332	turned	51	200	10200
and	2972	2	5944	you'll	30	300	9000
a	1775	3	5235	name	21	400	8400
he	877	10	8770	comes	16	500	8000
but	410	20	8400	group	13	600	7800
be	294	30	8820	lead	11	700	7700
there	222	40	8880	friends	10	800	8000
one	172	50	8600	begin	9	900	8100
about	158	60	9480	family	8	1000	8000
more	138	70	9660	brushed	4	2000	8000
never	124	80	9920	sins	2	3000	6000
Oh	116	90	10440	Could	2	4000	8000
two	104	100	10400	Applausive	1	8000	8000

Table 1: Empirical evaluation of Zipf's law on Tom Sawyer

Remark 1. The frequency of a term t in a collection \mathcal{D}_n of documents is defined as

$$\frac{|\{d \in \mathcal{D}_n \mid t \in d\}|}{n}.$$

The expected frequency of the term t_i is equal to $1/i$. At the same time, the expected frequency rank for t_i is exactly the i -th value among those of all terms. So the Zipf model reflects in a natural way Zipf's law. Since some of our motivating applications also deal with natural language texts, we can state that the Zipf model agrees with real life at least by degree distribution.

Remark 2. By defining the probability of the term t_i to be contained in a document as $1/i$, the set \mathcal{D} yields a probability space where a document d is an event that occurs with probability $P(d) = (\prod_{t_i \in d} \frac{1}{i}) (\prod_{t_i \notin d} 1 - \frac{1}{i})$.

In the following proofs we will use two inequalities ($a, b > 0$):

$$\left(1 - \frac{a}{b}\right)^b < e^{-a}, \quad a \leq b \quad (2.1)$$

$$\left(1 - \frac{1}{ab}\right)^a \geq 1 - \frac{1}{b}, \quad a, b \geq 1 \quad (2.2)$$

Indeed, let $g(x) = \ln(1-x)/x$, $0 < x < 1$. Notice that g is a decreasing function. The first inequality follows from $g(a/b) \leq \lim_{x \rightarrow 0} g(x) = -1$, while the second one is equivalent to $g(1/b) \leq g(1/ab)$.

For further considerations we introduce the following terms and definitions:

Definition 2.1. Let

$$\underbrace{t_1 t_2}_{P_1} \quad \underbrace{t_3 t_4 t_5 t_6 t_7}_{P_2} \quad \dots$$

be a partition of the set of terms. The group P_i includes terms from $t_{\lceil e^{i-1} \rceil}$ to $t_{\lfloor e^i \rfloor}$. We say that a document $d \in \mathcal{D}$ is *regular* if it contains exactly $\ln m$ terms $p_1 \dots p_{\ln m}$ such that $p_i \in P_i$.

Remark 3. Note that the expected number of terms in each group P_i is approximately one. If $\ln m$ is not an integer then the index of the last group is $\lceil \ln m \rceil$. In this case the expected number of terms in this group is smaller than one. To make the following proofs more legible we do not demand that the number of terms of a document or the matching size is an integer. In real settings these values have to be rounded appropriately.

Definition 2.2. Let $0 < \delta < 1$. We say that a document $d \in \mathcal{D}$ is δ -*n-generic* if the following holds:

$$\forall i \geq \delta \sqrt{2 \ln n} : \quad |\{t_j \in d \mid j \leq e^i\}| \geq (1 - \delta)i.$$

Lemma 2.3. Let $0 < \delta < 1$ and $c = e^{-\delta^2/2}$. Let $d \in \mathcal{D}$ be a random document following the Zipf model. It holds that for a sufficiently large n the probability that d is δ -*n-generic* is greater than $1 - c^{1.4\delta\sqrt{\ln n}}/(1 - c)$.

Proof. Let X be a random variable denoting the expected number of terms in d up to the term t_{e^i} . For a fixed i the Chernoff bound $P(X \leq (1 - \delta)EX) \leq e^{-EX\delta^2/2}$ yield that the probability that d contains less than $(1 - \delta)i$ terms up to the term t_{e^i} is smaller than $e^{-i\delta^2/2}$. This holds since $i < EX$ and therefore

$$P(X \leq (1 - \delta)i) \leq P(X \leq (1 - \delta)EX) \leq e^{-EX\delta^2/2} < e^{-i\delta^2/2}.$$

So the probability that d is not δ -*n-generic* is for large n bounded by

$$\begin{aligned} \sum_{i \geq 0} c^i - \sum_{i=0}^{\lfloor \delta \sqrt{2 \ln n} \rfloor - 1} c^i &= \frac{1}{1 - c} - \frac{1 - c^{\lfloor \delta \sqrt{2 \ln n} \rfloor}}{1 - c} \\ &= \frac{c^{\lfloor \delta \sqrt{2 \ln n} \rfloor}}{1 - c} \\ &\leq \frac{c^{\sqrt{2} \delta \sqrt{\ln n} - 1}}{1 - c} \\ &< \frac{c^{1.4 \delta \sqrt{\ln n}}}{1 - c} \end{aligned}$$

This holds because $1.4 < \sqrt{2}$ and n is large. Overall, the probability that d contains $(1 - \delta)i$ or more terms is greater than $1 - c^{1.4\delta\sqrt{\ln n}}/(1 - c)$. \square

Lemma 2.4. *Let $d \in \mathcal{D}$ be a random document following the Zipf model and let $0 < \delta < 1$ and $c = e^{-\delta^2/2}$. If we insert the first $\delta\sqrt{2\ln n}$ missing terms to d (assuming that there are missing terms), then for a sufficiently large n the following holds:*

$$P\left(\forall i \leq \sqrt{2\ln n} : |\{t_j \in d \mid j \leq e^i\}| \geq i\right) > 1 - c^{1.4\delta\sqrt{\ln n}}/(1 - c).$$

Proof. Since we insert the first $\delta\sqrt{2\ln n}$ missing terms to d , the number of terms in d is always at least $\delta\sqrt{2\ln n}$. Thus, the statement holds trivially for $i < \delta\sqrt{2\ln n}$. So let's consider the case $i \geq \delta\sqrt{2\ln n}$. By Lemma 2.3 we know that the probability that d is δ - n -generic is greater than $1 - c^{1.4\delta\sqrt{\ln n}}/(1 - c)$ for large n . Now, by inserting the first $\delta\sqrt{2\ln n}$ missing terms to d , we see that the probability that there are at least i terms t_j with $j \leq e^i$ is greater than $1 - c^{1.4\delta\sqrt{\ln n}}/(1 - c)$. \square

We now introduce a threshold, called *matching level*, to give statements about the most probable size of a maximal intersection:

$$q = q_n := \sqrt{2\ln n}.$$

Theorem 2.5 (Matching Level for the Zipf Model). *Let $\mathcal{D}_n = \{d_1, \dots, d_n\}$ be a document collection following the Zipf model.*

1. (Prefix match). *Let $0 < \delta < 1$ be fixed. Let $\gamma = 2 + \delta\sqrt{2\ln n}$ and $c = e^{-\delta^2/2}$. For sufficiently large n, m the following holds: The probability that there exists a document in \mathcal{D}_n that contains the first $q - \gamma$ terms of a query document $d_{\text{new}} \in \mathcal{D}$ following the Zipf model is greater than $1 - c^{\delta\sqrt{\ln n}}/(1 - c)$. Thus, the probability tends to one as $n \rightarrow \infty$.*
2. (Any match). *Let $\varepsilon > 0$ be fixed. The probability that there exists a document in \mathcal{D}_n that contains more than $(1 + \varepsilon)q$ terms of a query document $d_{\text{new}} \in \mathcal{D}$ following the Zipf model tends to zero as $n \rightarrow \infty$.*

Proof. 1. Let d_R be a fixed regular document (Definition 2.1). The probability that a document from \mathcal{D}_n contains the prefix of length $q - 2$ of d_R is at least

$$\begin{aligned} \frac{1}{e} \cdots \frac{1}{e^{q-2}} &> \frac{1}{e^{(q-1)^2/2}} = \frac{1}{e^{(q^2-2q+1)/2}} \\ &= \frac{e^{q-1/2}}{n}. \end{aligned}$$

Note that $e^{q-1/2} < n$ since $e^{(q-1)^2/2} > 1$. This means that the probability that there exists no document in \mathcal{D}_n that contains the $(q - 2)$ -prefix of d_R is no more than

$$\left(1 - \frac{e^{q-1/2}}{n}\right)^n < e^{-e^{q-1/2}},$$

which follows from inequality (2.1). So with probability greater than $1 - e^{-e^{q-1/2}}$ there exists a document in \mathcal{D}_n that has all terms from the $(q-2)$ -prefix of d_R . Consider d_{new} . If we insert the first $\delta\sqrt{2\ln n}$ missing terms to d_{new} , Lemma 2.4 implies that for large n the probability that d_{new} contains in every group P_i , $i \leq \sqrt{2\ln n}$, at least as many terms as d_R is greater than $1 - c^{1.4\delta\sqrt{\ln n}}/(1-c)$. Therefore, the probability that there exists a document in \mathcal{D}_n that matches the prefix of length $q-2$ of the extended query document d_{new} is greater than

$$\left(1 - c^{1.4\delta\sqrt{\ln n}}/(1-c)\right) \left(1 - e^{-e^{q-1/2}}\right).$$

For large n , this product is at least $1 - c^{\delta\sqrt{\ln n}}/(1-c)$. It remains to notice that by removing the initially inserted $\delta\sqrt{2\ln n}$ terms from d_{new} we still match $q-\gamma$ terms. This concludes the proof.

2. Let us fix a query document d_{new} for now. Let d be a random document following the Zipf model and let $\lambda > 0$. We can evaluate the Laplace transform of the intersection size as follows:

$$\begin{aligned} E \exp(\lambda |d_{new} \cap d|) &= \prod_{j: t_j \in d_{new}} \left(1 - \frac{1}{j} + \frac{1}{j} e^\lambda\right) \leq \prod_{j: t_j \in d_{new}} \left(1 + \frac{e^\lambda}{j}\right) \\ &= \prod_{\substack{j: t_j \in d_{new} \\ j > e^\lambda}} \left(1 + \frac{e^\lambda}{j}\right) \cdot \prod_{\substack{j: t_j \in d_{new} \\ j \leq e^\lambda}} \frac{e^\lambda}{j} \left(1 + \frac{j}{e^\lambda}\right). \end{aligned}$$

It follows that

$$\begin{aligned} \ln E \exp(\lambda |d_{new} \cap d|) &\leq \sum_{\substack{j: t_j \in d_{new} \\ j \geq e^\lambda}} \frac{e^\lambda}{j} + \sum_{\substack{j: t_j \in d_{new} \\ j \leq e^\lambda}} (\lambda - \ln j) + \sum_{\substack{j: t_j \in d_{new} \\ j \leq e^\lambda}} \frac{j}{e^\lambda} \\ &= e^\lambda T_1 + \lambda T_2 - T_3 + e^{-\lambda} T_4, \end{aligned}$$

where

$$\begin{aligned} T_1 &= \sum_{\substack{j: t_j \in d_{new} \\ j \geq e^\lambda}} \frac{1}{j}, & T_2 &= |d_{new} \cap [1, e^\lambda]|, \\ T_3 &= \sum_{\substack{j: t_j \in d_{new} \\ j \leq e^\lambda}} \ln j, & T_4 &= \sum_{\substack{j: t_j \in d_{new} \\ j \leq e^\lambda}} j. \end{aligned}$$

We will assume that our fixed query d_{new} satisfies the following four *regularity conditions*.

$$\begin{aligned} e^\lambda T_1 &\leq \varepsilon \lambda^2, & T_2 &\leq (1 + \varepsilon) \lambda, \\ T_3 &\geq (1 - \varepsilon) \frac{\lambda^2}{2}, & e^{-\lambda} T_4 &\leq \varepsilon \lambda^2. \end{aligned}$$

Under these regularity conditions we obtain

$$\ln E \exp(\lambda |d_{new} \cap d|) \leq (1 + 7\varepsilon) \frac{\lambda^2}{2}.$$

Assuming (d_j) to be a sample of n independent documents distributed according to the Zipf model, by the Chebyshev exponential inequality, for any $r > 0$ we have

$$\begin{aligned} P\left(\max_{j \leq n} |d_{new} \cap d_j| \geq r\right) &\leq n P(|d_{new} \cap d| \geq r) \\ &\leq n \frac{E \exp(\lambda |d_{new} \cap d|)}{e^{\lambda r}} \\ &\leq n \exp\left((1 + 7\varepsilon) \frac{\lambda^2}{2} - \lambda r\right). \end{aligned}$$

Take any $\gamma > 0$. By choosing now

$$r = (\sqrt{2 \ln n} + \gamma)(1 + 7\varepsilon)$$

and

$$\lambda = \sqrt{2 \ln n} + \gamma$$

we obtain $(1 + 7\varepsilon)\lambda^2 = \lambda r$, hence

$$\begin{aligned} P\left(\max_{j \leq n} |d_{new} \cap d_j| \geq r\right) &\leq n \exp\left(-(1 + 7\varepsilon) \frac{\lambda^2}{2}\right) \\ &\leq n \exp\left(-\frac{(\sqrt{2 \ln n} + \gamma)^2}{2}\right) \rightarrow 0 \end{aligned}$$

for $n \rightarrow \infty$, as required. Let now the query d_{new} be randomly chosen according to the Zipf model. For a sufficiently large λ it is true that

$$\begin{aligned} ET_1 &= \sum_{j \geq e^\lambda} \frac{1}{j^2} \leq e^{-\lambda}; \\ ET_2 &= \sum_{j \leq e^\lambda} \frac{1}{j} \leq \lambda; \quad VarT_2 \leq \sum_{j \leq e^\lambda} \frac{1}{j} \leq \lambda; \\ ET_3 &= \sum_{j \leq e^\lambda} \frac{\ln j}{j} \leq \frac{\lambda^2}{2}; \quad VarT_3 \leq \sum_{j \leq e^\lambda} \frac{\ln^2 j}{j} \leq \frac{\lambda^3}{3}; \\ ET_4 &= \sum_{j \leq e^\lambda} 1 \leq e^\lambda. \end{aligned}$$

Let us explain the bound for ET_3 . Let $2 \leq b \leq S$ be integers.

$$\sum_{j \leq S} \frac{\ln j}{j} \leq \int_b^S \frac{\ln x}{x} dx + \sum_{j=2}^b \frac{\ln j}{j} = \frac{\ln^2 S}{2} - \frac{\ln^2 b}{2} + \sum_{j=2}^b \frac{\ln j}{j}.$$

For $b \geq 21$ it holds that

$$\frac{\ln^2 b}{2} \geq \sum_{j=2}^b \frac{\ln j}{j}$$

and therefore

$$\sum_{j \leq S} \frac{\ln j}{j} \leq \frac{\ln^2 S}{2}.$$

Hence, $\lambda \geq 4$ yields the desired bound on ET_3 . The bound on $VarT_3$ is shown by similar techniques.

It remains to notice that the probability of each of the four regularity conditions to be true for d_{new} tends to one as $\lambda \rightarrow \infty$. Indeed, the expectations and variances calculated above easily show this fact. \square

By Theorem 2.5 we can conclude that with high probability there exists a document in \mathcal{D} that matches the prefix of length $q - \gamma$ of d_{new} , whereas the probability to find a document that has more than $(1 + \varepsilon)q$ common terms with d_{new} (at arbitrary positions) is quite small. Therefore, it suffices to determine a document that has a maximal common prefix with the query document. This fact, however, allows to sort the documents according to their sorted term lists² and then perform a binary search based on the sorted term list of the query document (Figure 2). The running time is as follows (for the *average* case³

Preprocessing:

1. For every document: Sort the term list according to the position of the terms in the frequency table.
2. Sort the documents according to their sorted term lists.

Query: Find a document having the maximal common prefix with the query document by binary search.

Figure 2: **MaxInt algorithm in the Zipf model**

analysis we assume that the length of term lists is $\log m \in \mathcal{O}(\log n)$, for the *worst* case analysis the length is m):

²In a sorted term list the terms of a document are ordered according to the position of the terms in the frequency table.

³Only for the average case our constraints from Section 1 are preserved.

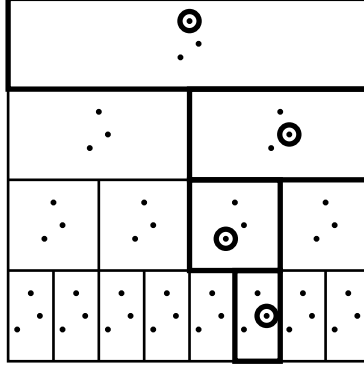


Figure 3: Hierarchical scheme

	average case	worst case
Step 1	$\mathcal{O}(n \cdot \log n \cdot \log \log n)$	$\mathcal{O}(n \cdot m \cdot \log n)$
Step 2	$\mathcal{O}(n \cdot \log^2 n)$	$\mathcal{O}(n \cdot \log n \cdot m)$
Query	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(\log n \cdot m)$

The log factor in the query step is due to the fact that the algorithm performs a binary search on the document collection. Since in the average case the length of a term list is $\mathcal{O}(\log n)$, we get another log factor resulting in a query time of $\mathcal{O}(\log^2 n)$. One can try to improve the accuracy of our algorithm by finding a “maximal prefix with at most one difference to the query document”. A recent technique called “indexing with errors” [6, 16] might be useful for such an extension.

3 MaxInt in the Hierarchical Scheme

Our second model is motivated by the observation that in many existing applications terms can be ordered hierarchically. Let $k \geq 8$ be an integer and let \mathcal{T} be a set of $(2^k - 1) \cdot k$ different terms. A document collection \mathcal{D}_k consists of 2^k sets where every set $d \in \mathcal{D}_k$ is a subset of \mathcal{T} with $|d| = k$. A *hierarchical scheme* is a table with k levels, level 1 to level k . Level i , $1 \leq i \leq k$, is divided into 2^{i-1} cells, cell $C_{i,1}$ to cell $C_{i,2^{i-1}}$. For $2 \leq l \leq k$ we say that cell $C_{l-1,j}$, $1 \leq j \leq 2^{l-2}$, is *above* cell $C_{l,j'}$, $1 \leq j' \leq 2^{l-1}$, if $\lceil j'/2 \rceil = j$. Every cell contains k terms. A document collection based on this scheme can be generated as follows: Every document is generated independently. Choose a random cell on level k and mark it. Then, for $l = \{k, \dots, 2\}$, mark on level $l - 1$ the cell that is above the already marked cell on level l . Now choose one random term in every marked cell. The so defined set of terms form a document of our collection. Note that every document generated by this process corresponds to a unique sequence of cells. We’ll call such a sequence a *cell path*. There exists a natural ordering on

these cell paths where the cells $C_{1,1}, C_{2,1}, \dots, C_{k,1}$ describe the leftmost path, the cells $C_{1,1}, C_{2,2}, \dots, C_{k,2^{k-1}}$ accordingly the rightmost one (see Figure 3).

Remark 4. We claim that the hierarchical scheme follows *Zipf's law*. To be more precise, the following holds: For every level the product of expected frequency and expected frequency rank of a term is the same. Indeed, the expected frequency of a term on level i is given by the formula $2^k / (2^{i-1} \cdot k)$. The expected rank of a term is given by the formula $(2^{i-1} - 1) \cdot k + 2^{i-2} \cdot k$. Hence, the product between frequency and frequency rank (divided by 2^k) is equal to

$$\frac{2^k}{2^{i-1} \cdot k} \cdot \left(\frac{3}{2} \cdot 2^{i-1} - 1 \right) \cdot \frac{k}{2^k} \in [0.5, 1.5],$$

which means it lies in a fixed interval and therefore follows the idea of Zipf's law.

This time we introduce two *matching levels* to give statements about the most probable size of a maximal intersection. The matching levels are

$$q = \frac{k}{1 + \log k} \quad \text{and} \quad q' = \frac{k}{\log k}.$$

Remark 5. Again, to keep proofs more legible, we do not demand that the length of a prefix or the matching size is an integer (except part one of the following theorem). But clearly, in real settings these values have to be rounded appropriately.

Theorem 3.1 (Matching Levels for Hierarchical Scheme). *Let $k \geq 8$ be an integer and $2 \leq \gamma < q$. Let \mathcal{D}_k be a document collection following the hierarchical scheme.*

1. (Prefix match). *The probability that there exists a document $d \in \mathcal{D}_k$ that matches the first $\lfloor q - \gamma \rfloor$ terms (i.e. the terms from level 1 to level $\lfloor q - \gamma \rfloor$) of a query document d_{new} following the hierarchical scheme is greater than $1 - 2^{-(2k)^\gamma}$.*
2. (Any match). *The probability that there exists a document $d \in \mathcal{D}_k$ that matches at least $q' + \gamma$ terms of a query document d_{new} following the hierarchical scheme is smaller than $2/k^{\gamma-1}$.*

Proof. 1. The number of different prefixes of length $\lfloor q - \gamma \rfloor$ is at most

$$k(2k)^{q-\gamma-1} < 2^{(1+\log k)(q-\gamma)} = 2^{(1+\log k)(k/(1+\log k)-\gamma)} = 2^k \cdot (2k)^{-\gamma}.$$

So the probability that a new document does not match a prefix of length $\lfloor q - \gamma \rfloor$ with any document from \mathcal{D}_k is smaller than

$$\left(1 - \frac{(2k)^\gamma}{2^k} \right)^{2^k} < e^{-(2k)^\gamma} < 2^{-(2k)^\gamma}.$$

For $k \geq 8$ and $\gamma < q$ it holds that $(2k)^\gamma < 2^k$ and therefore the above inequality follows from inequality (2.1). We get that the probability that there exists a document in \mathcal{D}_k with the same prefix as d_{new} of length $\lfloor q - \gamma \rfloor$ is greater than $1 - 2^{-(2k)^\gamma}$.

2. Let $t \geq q' + \gamma$ be the last position where the terms of d and d_{new} match. We want to estimate the probability that d_{new} matches at least $q' + \gamma$ terms at arbitrary positions with d . The probability that the first t terms (beginning at level 1) of d and d_{new} are all in the same cells is 2^{1-t} . The probability that at least $q' + \gamma$ terms are matched on some fixed positions is no more than $(1/k)^{q'+\gamma} \cdot ((k-1)/k)^{t-q'-\gamma}$. An upper bound for the number of different possibilities of matching at least $q' + \gamma$ out of t terms is 2^t . Since the factor $((k-1)/k)^{t-q'-\gamma}$ is smaller than 1, overall we get that the probability that d_{new} matches at least $q' + \gamma$ terms at arbitrary positions with d is smaller than

$$k \cdot 2^t \cdot \left(\frac{1}{k}\right)^{q'+\gamma} \cdot 2^{1-t} = 2 \cdot k \cdot \left(\frac{1}{k}\right)^{q'+\gamma} = 2 \cdot \left(\frac{1}{k}\right)^{q'+\gamma-1}.$$

The factor k in the above equation is due to the fact that we need to consider all possible levels for the last matched position t . Now the probability that no document matches at $q' + \gamma$ arbitrary positions with d_{new} is at least

$$\left(1 - 2 \cdot \left(\frac{1}{k}\right)^{q'+\gamma-1}\right)^{2^k} = \left(1 - \frac{1}{2^k \cdot \frac{k^{\gamma-1}}{2}}\right)^{2^k} \geq 1 - \frac{2}{k^{\gamma-1}},$$

which follows from inequality (2.2), since $\gamma \geq 2$ and $k \geq 8$. So the probability that there exists a document in \mathcal{D}_k that matches at least $q' + \gamma$ terms of d_{new} is smaller than $2/k^{\gamma-1}$. □

Theorem 3.1 yields that also for the hierarchical scheme it suffices to search a document that has a maximal common prefix with the query document. The resulting algorithm is analogue to the one for the Zipf model and summarized on Figure 4.

The running time is shown in the table below. Note that for the hierarchical scheme we only perform a worst case analysis since every document has equal length.

	average case	worst case
Step 1	—	$\mathcal{O}(2^k \cdot k \cdot \log k)$
Step 2	—	$\mathcal{O}(2^k \cdot k^2)$
Query	—	$\mathcal{O}(k^2)$

Preprocessing:

1. For every document: Sort the term list according to the hierarchical scheme, i.e. according to the levels in which the terms appear.
2. Sort the documents according to their corresponding cell paths, i.e. documents that correspond to the leftmost path in the scheme are at the beginning of the sorted list. Documents that correspond to the same cell path are sorted lexicographically.

Query: Find a document having the maximal common prefix with the query document by binary search.

Figure 4: **MaxInt algorithm in the hierarchical scheme**

4 Further Work

In this paper we have shown that assumptions on the random nature of the input can lead to *provable* time and accuracy bounds for MAXINT. Also, we have discovered a MAXINT threshold phenomenon in two randomized models.

The next step is to understand it better. Does it hold for other randomized models from [18], especially for generalized random graphs with a power-law degree distribution? Does it hold in the real life networks? Can we introduce randomized models for sparse vector collections and find a similar effect there? It was observed that tractable instances of nearest neighbors have small intrinsic dimension [5, 8, 14, 12]. Does the same effect hold for the Zipf model and the hierarchical scheme? Of course, the most challenging problem is to find an exact algorithm for MAXINT (preserving our time constraints) or to prove its hardness. What are other particular cases or assumptions that have efficient MAXINT solutions? On the other hand, we have a very particular subcase for which we still do not believe in a positive solution. Hence, we ask for a hardness proof for the following *on-line inclusion problem*.

On-line Inclusion Problem

Database: A family \mathcal{F} of 2^k subsets of $[1 \dots k^2]$.

Query: Given a set $f_{new} \subseteq [1 \dots k^2]$, decide whether there exists an $f \in \mathcal{F}$ such that $f_{new} \subseteq f$.

Constraints: Space for preprocessed data $2^k \cdot \text{poly}(k)$.
Query time $\text{poly}(k)$.

Note that we have a constraint on space for preprocessing, not time. A related

problem but with a much stronger restriction on preprocessing space was proven to be hard by Bruck and Naor [3].

Our algorithm in Section 3 uses polylogarithmic time (in the number of documents) but it returns only an approximate solution with high probability (not every time). Can we get an optimal solution or at least a *guaranteed* approximation by relaxing the time constraint to *expected* polylogarithmic time?

The maximal intersection problem is a special case of a whole family of problems called *Strongest Connection Problems* (SCP) which covers all problems fitting the following framework. Consider some class of graphs \mathcal{G} and some class of paths \mathcal{P} .

Strongest Connection Problems

Database: A graph $G \in \mathcal{G}$.

Query: Given a (new) vertex v (together with edges connecting v with G), return a vertex $u \in G$ with a maximal number of \mathcal{P} -paths from v to u .

Constraints: Preprocessing time $o(|G|^2)$.
Query time $o(|G|)$.

A number of well-motivated problems fall into the family of SCP. Some of them are listed below but the application of the SCP framework is not limited to these instances.

Recommendations. \mathcal{G} : bipartite graphs; \mathcal{P} : paths of length three.

Example: A graph G is partitioned into vertices representing people and books, and the edge relation describes who has bought which books. Given a person v . Which is a book (not purchased by v) that is most often bought by people that had been interested in books of v ?

Similarity in folksonomies. \mathcal{G} : tripartite 3-graphs where the set of vertices is partitioned into (V_0, V_1, V_2) and E is an edge relation in $V_0 \times V_1 \times V_2$; \mathcal{P} : paths consisting of two edges that overlap either in V_0 or in V_1 or both in V_0 and V_1 .

Example: A graph representing all events of kind “a user U used a tag T to label a website W ”. Then, the strongest connection query for the tag T_{new} returns another tag that was most often co-used by the same users and/or applied to the same websites. Such tripartite 3-graph is accumulated in `del.icio.us`.

Acknowledgments. We thank the anonymous referees for their very detailed reports. We also thank Volker Diekert, Jörn Laun, and Ulrich Hertrampf for many helpful comments.

References

- [1] C. Böhm, S. Berchtold, and D. A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Comput. Surv.*, 33(3):322–373, 2001.
- [2] A. Broder. On the resemblance and containment of documents. In *SEQUENCES '97: Proceedings of the Compression and Complexity of Sequences 1997*, page 21, Washington, DC, USA, 1997. IEEE Computer Society.
- [3] J. Bruck and M. Naor. The hardness of decoding linear codes with preprocessing. *IEEE Transactions on Information Theory*, 36(2):381–385, 1990.
- [4] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín. Searching in metric spaces. *ACM Comput. Surv.*, 33(3):273–321, 2001.
- [5] K. Clarkson. Nearest-neighbor searching and metric space dimensions. In *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*, MIT Press, 2006.
- [6] R. Cole, L.-A. Gottlieb, and M. Lewenstein. Dictionary matching and indexing with errors and don't cares. In *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 91–100, New York, NY, USA, 2004. ACM.
- [7] R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 301–312, New York, NY, USA, 2003. ACM.
- [8] N. Goyal, Y. Lifshits, and H. Schütze. Disorder inequality: a combinatorial approach to nearest neighbor search. In *WSDM*, pages 25–32, 2008.
- [9] G. R. Hjaltason and H. Samet. Index-driven similarity search in metric spaces (survey article). *ACM Trans. Database Syst.*, 28(4):517–580, 2003.
- [10] B. Hoffmann, Y. Lifshits, and D. Nowotka. Maximal intersection queries in randomized graph models. In V. Diekert, M. V. Volkov, and A. Voronkov, editors, *CSR*, volume 4649 of *Lecture Notes in Computer Science*, pages 227–236. Springer, 2007.
- [11] P. Indyk. Nearest neighbors in high-dimensional spaces. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry, chapter 39*. CRC Press, 2004. 2nd edition.
- [12] D. R. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. In *STOC '02: Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pages 741–750, New York, NY, USA, 2002. ACM.

- [13] J. M. Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 599–608, New York, NY, USA, 1997. ACM.
- [14] R. Krauthgamer and J. R. Lee. Navigating nets: simple algorithms for proximity search. In *SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 798–807, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.
- [15] Y. Lifshits and D. Nowotka. Estimation of the click volume by large scale regression analysis. In V. Diekert, M. V. Volkov, and A. Voronkov, editors, *CSR*, volume 4649 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 2007.
- [16] M. G. Maaß and J. Nowak. A new method for approximate indexing and dictionarylookup with one error. *Inf. Process. Lett.*, 96(5):185–191, 2005.
- [17] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.
- [18] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167, 2003.
- [19] P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity Search - The Metric Space Approach*, volume 32. Springer, 2006.
- [20] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Comput. Surv.*, 38(2):6, 2006.